UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/050,387	01/16/2002	Nicolai Kosche	004-7051	6183

22120 7590 05/31/2005

ZAGORIN O'BRIEN GRAHAM LLP
7600B N. CAPITAL OF TEXAS HWY.
SUITE 350
AUSTIN, TX 78731

EXAMINER	
VO, TED T	

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 05/31/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/050,387	KOSCHE ET AL.
	Examiner	Art Unit
	Ted T. Vo	2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 February 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-69* is/are pending in the application.

4a) Of the above claim(s) 54-60 is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-53 and 61-69* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☒ Claim(s) *54-60* are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 02/22/05.

Claims 33 is amended. Claims 54-60 are withdrawn.

In view of the amendment, the objection to the specification and the rejection of Claim 33 under 35 U.S.C. 112, second paragraph, are withdrawn.

Claims 1-69 are pending in the application.

Election/Restrictions

2. Applicant's election with traverse of claims 1-33, 34-53, 61-69, and Applicants withdraw Claims 54-60 in the reply filed on 02/22/05 is acknowledged. Applicants' arguments give no reasons why they traversed. The requirement is still deemed proper and is therefore **made FINAL**.

This application contains claims 54-60 are drawn to an invention nonelected with traverse in Office Action dated 10/21/04, where replying to the Office Action, Applicants give no reasons. A complete reply to the final rejection must include cancellation of nonelected claims or other appropriate action (37 CFR 1.144) See MPEP § 821.01.

Response to Arguments

3. Applicants' arguments in the Remarks section filed on 02/22/2005 have been fully considered. However, the arguments are not persuasive.

Particularly, Applicants stated, "All of the rejection either are devoid of any disclosure of any Applicants' limitations and reply solely on Applicants' own claim language, or improperly construes Applicant's claims". Applicants list the passage "*When bytecodes are first loaded, they are run through the interpreter. The profiler keeps a record of runtimes for each method. When a method is found to be* *taking a lot of time, HotSpot compiles and optimizes it. Every future call to that method uses the native machine instructions produced by the compiler*", and mention the HotSpot of pages 3 and 4, Figure 3 are "flaw prevalent" and insufficient (Remarks: page 15).

Examiner respectfully disagrees: The breadth of the claims is broad and the claimed language abstractly recites generic functionality of an execution. The Claims are thus interpretable and disclosed by a profiler embedded in a Sun's next-generation Dynamic Compiler provided to a Virtual machine for Hotspot, as discussed by Armstrong. While Applicants still maintain the broad breadth of the Claims, Applicants' arguments are merely that the Sun's next-generation Dynamic Compiler mentioned by Armstrong does not disclose or suggest their broadly recitations. Applicants do not respond properly to the Examiner's citations, particularly, pointing to the functions performed by Sun's next-generation Dynamic Compiler. It should be noted that the major recitations in the Claims are broad and are the common features, such as "executable instance" "characteristic", "operation", "cache miss likehood", "binary translator", "JIT Compiler", etc., performed in a compiler, particularly, in a JIT virtual machine. Thus, the functionality of Hotspot and profiler used in Sun's next-generation Dynamic Compiler in a JIT Virtual machine cited in the Office Action is properly addressing the broadly claimed recitations. Within the Applicants' remarks, Examiner respectfully addresses that there is no proper response with respect to limitations/citations addressed in the action, but instead Applicants content such Examiner's citations are "flaw prevalent". Thus, Examiner will maintain that the Sun's next-generation Dynamic Compiler discussed by Armstrong and the whole reference disclose the claimed limitations; the given citations are proper to the broad breadth of the Claims.

For example, consider the Claim 1:

Claim language: *identifying at least one operation in first executable instance of code*

Identifying one operation is broad. Such broad recitation is suggested in the reference such as "When a method is found" (Page 4, second paragraph).

Claim language: *executing the first executable instance and responsive to detection of an execution event, associating a corresponding execution characteristic with a corresponding identified one of the operations*

The recitation of the execution of the first executable instance is broad. The profiler/dynamic compiler under the below passage performs such broad execution (The passages in first and second paragraph of page 4),

"When bytecodes are first loaded, they are run through the interpreter. The profiler keeps a record of runtimes for each method ('*executing of the first executable instance*'). When a method is found (*identifying at least one operation)* to be taking a lot of time, HotSpot compiles and optimizes it. Every future call to that method ('*executing the first executable instance*') uses the native machine instructions produced by the compiler.
As with a JIT, the results of the compilation are not kept between runs ('*executing*'). Because the bytecodes are more compact, this saves loading time as well as storage space. It also retains portability, and allows the optimization to reflect the way the program is currently being used. Currently, no attempt is made to save information that might help future runs become efficient more quickly. Although this approach may be a future possibility, Sun has no such plans at this time."

It is clearly these passages disclose the same functionality of the claimed recitation.

<u>Claim language</u>: *preparing a second executable instance of the code based, at least in part, on the association between the execution characteristic and the identified operation*

The recitation of *preparing a second executable instance of the code based* is broad. Sun's Dynamic compiler discloses preparing of new instructions for resolving hotspot by referring "Every future call to that method uses the <u>native machine instructions produced by the compiler</u>". This statement clearly addresses such claimed recitation preparing.

With regards the Applicants' arguments of the rejection of Claims 34-39, 41-45, 53, 61-69. These Claims recite the optimizer compiler, the computer program product, and the apparatus, respectively. Applicants alleged the rejection of these claims is merely referred to other rejections. It should be note that if the Claims are having the functionality of other rejected Claims, the referring is proper.

For example, consider Claim 34's recitation:

"*An optimizing compiler that prepares a second executable instance of computer program code including optimizations in addition to those of a previously prepared first executable instance thereof, wherein the additional optimizations include performing one or more transformations based on run-time information from execution of the first executable instance, wherein consistency of instruction identification is maintained from preparation of the first executable instance to preparation of the second executable instance.*", and same manners as recited in claim 61 and 66. Such limitations are corresponding to the recitation of Claim 1, where the hotspot detected by the profiler prepares a second execution as shown in the passages above to suggest the broad breadth of the claim.

Because the claims are broad, the interpretation of the claims given as in the manner of the claimed language and the specification to the reference is proper. It is not flaw as applicants argued. Furthermore, through the Armstrong disclosure, the reference reads through the broad claimed languages of the dependent claims, particularly, Armstrong discussion of the Hotspot is in Figure 3. The Figure 3 shows Sun's Hotspot to detect that uses a profile to identify bytecode. The Figure 3 performs the all the execution as of a JIT virtual machine. And the Hot spot discloses the heart of the claim: *preparing "a second executable instance"* by using the "native machine instructions" for a future execution. The fact is that Hotspot identifies an operation that causes the bottle next in the execution, and optimizes and generates code itemized code "native machine code" to replace the hotspot as for the future execution as discussed in pages 3 and 4. Therefore, Applicants' statement in their arguments, "All of the rejection either are devoid of any disclosure of any Applicants' limitations and reply solely on Applicants' own claim language, or improperly construes Applicant's claims" is improper, and does not directly respond to the examiner's citations.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-53, 61-69 are rejected under 35 U.S.C. 102(b) as being anticipated by Armstrong, "HotSpot: A new breed of virtual machine", Java World, http://www.javaworld.com, 1998.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Armstrong discloses,

A code preparation method comprising:

identifying at least one operation in first executable instance of code (Page 4, second paragraph, "When bytecodes are first loaded,..": referring "When a method is found");

executing the first executable instance and responsive to detection of an execution event (page 4, Second paragraph, referring "runtime"), *associating a corresponding execution characteristic* (page 4, Second paragraph, referring "profiler") *with a corresponding identified one of the operations* (page 4, Second paragraph, referring "method"/"HotSpot")*; and*

preparing a second executable instance of the code based, at least in part, on the association between the execution characteristic and the identified operation (page 4, second paragraph, referring "Every future call to that method uses the native machine instructions produced by the compiler": *preparing a second executable instance*),

See pages 3 and 4, description of How the dynamic compiler works.

As per Claim 2: Armstrong discloses,

The method of claim 1, wherein the operation identification is consistent between the first executable instance and the preparation of the second executable instance.

See page 4, second paragraph, referring "Every future call (*is consistent between*) to that method (*first executable instance*) uses the native machine instructions (*the second executable instance*) produced by the compiler"

As per Claim 3: Armstrong discloses,

The method of claim 2, wherein the consistency of operation identification is maintained from preparation of the first executable instance to preparation of the second executable instance.

See page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler" and the whole description of description of How the dynamic compiler works.

As per Claim 4: Armstrong discloses,

The method of claim 1, wherein same unique identification numbers are assigned to corresponding operations of the first executable and the second executable.

See page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler" and the whole description of description of How the dynamic compiler works, where *same unique identification* has been read from the correspondence between a method that has hotspot and is executed in the first time and the native optimized code (prepared code) executed in the place of that method.

As per Claim 5: Armstrong discloses,

The method of claim 4, wherein the execution characteristic is associated with the unique identification number.

see description of How the dynamic compiler works, the correspondence between a method that has hotspot and is executed in the first time and the native optimized code (prepared code) executed in the place of that method.

As per Claim 6: Armstrong discloses,

The method of claim 4, wherein the unique identification numbers and their assignment to operations are maintained throughout any optimizations or code transformations performed in preparation of the first executable.

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", and the whole description of description of How the dynamic compiler works.

As per Claim 7: Armstrong discloses,

The method of claim 6, wherein the maintenance of the unique identification number assignments include further assigning the unique identification number to a copy when an operation is copied as part of a code transformation or optimization.

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", and the whole description of description of How the dynamic compiler works.

As per Claim 8: Armstrong discloses,

> *The method of claim 6, wherein the maintenance of the unique identification number assignments includes removing an assignment when the assigned operation is removed as part of a code transformation or optimization*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler". Thus, the Hotspot compiler, the optimization, has removed the assignment that calls the method, and assigned such a call to the optimized native code in its future execution.

As per Claim 9: Armstrong discloses,

> *The method of claim 1, wherein the associating of the corresponding execution characteristic includes encoding aggregated hardware event information in an extended definition of an instruction instance for use in the preparation of the second executable instance.*

as providing profiling in the HotSpot compiler of the figure in page 3, where profiling of Hotspot is known for using counters (hardware event information) to detect a low performance, threshold, caused by a hotspot. See pages 3-4, second paragraph, "the whole description of description of How the dynamic compiler works, and Profiling and compiling heuristics.

As per Claim 10: Armstrong discloses,

> *the method of claim 1, wherein the identified operation is a memory access instruction*

in the HotSpot compiler mechanism: referring native instructions.

As per Claim 11: Armstrong discloses, based on the broad limitation,

> *The method of claim 1, wherein the execution characteristic includes a cache miss likelihood.*

in the HotSpot compiler mechanism because *cache miss* also causes hotspot.

As per Claim 12: Armstrong discloses,

> *The method of claim 1, wherein the preparation includes inserting one or more prefetch operations in the code prior to the identified operation to exploit latency provided by servicing of a cache miss by the identified operation.*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", where "uses the native machine instructions" is an act of inserting and prefetching, and where the method that is replaced by native machine instructions has hotspot such as looping, cache miss etc.

<u>As per Claim 13</u>: Armstrong discloses,

> *The method of claim 1, further comprising: preparing the first executable instance.*

see page 4, second paragraph, referring the profiling within the method at the first time before the future execution of native machine instructions.

<u>As per Claim 14</u>: Armstrong discloses,

> *The method of claim 13, wherein the preparation of the first executable instance includes substantially all optimizations operative in the preparation of the second executable.*

see page 4, first and second paragraphs, referring the act in optimizing hotspots to produce the <u>native instructions</u> (*second executable*).

<u>As per Claim 15</u>: Armstrong discloses,

> *The method of claim 14, wherein execution of the first executable instance corresponds substantially with execution of an executable instance of code prepared without the identifying.*

see page 4, first and second paragraphs.

<u>As per Claim 16</u>: Armstrong discloses,

> *The method of claim 14, whereby execution of the first executable instance sufficiently corresponds to that in an expected execution environment, so that the execution characteristic is applicable to the preparation of the second executable.*

see pages 3-4, whole description of How dynamic compiler works.

<u>As per Claim 17</u>: Armstrong discloses,

> *The method of claim 13, wherein the preparation of the first executable instance forgoes certain optimizations performed, after use of the association between the execution characteristic and the identified instruction, by the further preparing.*

see pages 3-4, referring profiling, and whole description of How dynamic compiler works.

<u>As per Claim 18</u>: Armstrong discloses,

The method of claim 13, wherein the preparation of the first executable instance includes compilation of the code.

see page 4, referring Profiling and compiling heuristics.

As per Claim 19: Armstrong discloses,

The method of claim 1, wherein both the first and the second executable instances are compiled instances of the code.

see pages 3-4, whole description of How dynamic compiler works.

As per Claim 20: Armstrong discloses,

The method of claim 1, wherein the second executable instance is an optimization of the first executable instance.

see page 4, second paragraph.

As per Claim 21: Armstrong discloses,

The method of claim 1, wherein the preparing includes optimizations forgone in the first executable instance.

see page 4, Profiling and compiling heuristics.

As per Claim 22: Armstrong discloses,

The method of claim 1, wherein the preparation of the second executable instance includes optimizations forgone in preparation of the first executable instance.

see pages 3-4, whole description of How dynamic compiler works; see page 4, Profiling and compiling heuristics.

As per Claim 23: Armstrong discloses,

The method of claim 1, wherein at least the preparing is performed by an optimizing compiler.

see the figure in page 3.

As per Claim 24: Armstrong discloses,

The method of claim 1, wherein at least the preparing is performed by a binary translator.

see the figure in page 3.

As per Claim 25: Armstrong discloses,

The method of claim 1, wherein at least the preparing is performed by a binary rewriter.

see the figure in page 3.

As per Claim 26: Armstrong discloses,

The method of claim 1, wherein at least the preparing is performed by a binary optimizer.

see the figure in page 3.

As per Claim 27: Armstrong discloses,

The method of claim 1, wherein at least the preparing is performed by a just-in-time (JIT) compiler.

see the figure in page 3.

As per Claim 28: Armstrong discloses,

The method of claim 1, wherein the associating of the corresponding execution characteristic includes aggregating contributions of plural instances of the execution event.

see the figure in page 3.

As per Claim 29: Armstrong discloses,

The method of claim 1, wherein the associating of the corresponding execution characteristic includes backtracking from a point in the code that coincides with delayed detection of the execution event.

see the figure in page 3.

As per Claim 30: Armstrong discloses,

The method of claim 1, wherein the associating of the corresponding identified one of the operations includes reading or receiving a computer readable encoding of an event profile.

see the figure in page 3 and see page 4: Profiling and compiling heuristics.

As per Claim 31: Armstrong discloses,

The method of claim 1, wherein the associating of the corresponding execution characteristic includes reading or receiving a computer readable encoding of an event profile.

see the figure in page 3 and see page 4: Profiling and compiling heuristics.

As per Claim 32: Armstrong discloses,

The method of claim 1, further comprising: preparing the second executable instance as a computer program product for distribution, transmission or execution.

see the figure in page 3 and the computer program product is referred to Java program.

As per Claim 33: Armstrong discloses,

The method of claim 32, wherein the computer program product is encoded in one or more computer readable media selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.

see the figure in page 3 and the computer program product is referred to Java program.

As per Claim 34: Claim 34 recites a compiler that performs the steps corresponding to the limitation of Claim 1. See the rationale on Claim 1 above.

As per Claim 35: Claim 35 recites a compiler that performs the steps corresponding to the limitation of Claim 2. See the rationale on Claim 2 above.

As per Claim 36: Claim 36 recites a compiler that performs the steps corresponding to the limitation of Claim 6. See the rationale on Claim 6 above.

As per Claim 37: Claim 37 recites a compiler that performs the steps corresponding to the limitation of Claim 7. See the rationale on Claim 7 above.

As per Claim 38: Claim 38 recites a compiler that performs the steps corresponding to the limitation of Claim 8. See the rationale on Claim 8 above.

As per Claim 39: Claim 39 recites a compiler that performs the steps corresponding to the limitation of Claim 12. See the rationale on Claim 12 above.

As per Claim 40: Regarding limitation,

"*The optimizing compiler of claim 34, wherein the transformations include insertion of one or more non-faulting loads*", see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", where "uses the native machine instructions" is an act of inserting, and see "native machine instructions" (*non-faulting loads*).

As per Claim 41: Claim 41 recites a compiler that perfoms the steps corresponding to the limitation of Claim 13. See the rationale on Claim 13 above.

As per Claim 42: Claim 42 recites a compiler that perfoms the steps corresponding to the limitation of Claim 14. See the rationale on Claim 14 above.

As per Claim 43: Claim 43 recites a compiler that perfoms the steps corresponding to the limitation of Claim 11. See the rationale on Claim 11 above.

As per Claim 44: Claim 44 recites a compiler that perfoms the steps corresponding to the limitation of Claim 10. See the rationale on Claim 10 above.

As per Claim 45: Claim 45 recites a compiler that perfoms the steps corresponding to the limitation of Claim 14. See the rationale on Claim 14 above.

As per Claim 46: Regarding,

The optimizing compiler of claim 34, embodied as part of a binary translator

referring to the figure in page 3.

As per Claim 47: Regarding,

The optimizing compiler of claim 34, embodied as part of a binary rewriter.

referring to the figure in page 3.

As per Claim 48: Regarding,

The optimizing compiler of claim 34, embodied as part of a binary optimizer.

referring to the figure in page 3.

As per Claim 49: Regarding,

The optimizing compiler of claim 34, embodied as a just-in-time (JIT) compiler.

referring to the figure in page 3.

As per Claim 50: Claim 50 recites a compiler that performs the steps corresponding to the limitation of Claim 6. See the rationale on Claim 6 above.

As per Claim 51: Regarding,

> *The optimizing compiler of claim 34, wherein the optimizing compiler identifies one or more memory access instructions in the first executable instance of the computer program code; and wherein the run-time information encodes respective execution characteristics for respective ones of the identified memory access instructions.*

see pages 4-5, referring to Profiling and compiling heuristics.

As per Claim 52: Regarding,

> *The optimizing compiler of claim 34, wherein collection of the run-time information includes aggregation of execution event information and association of the aggregated information with memory access instructions identified in the first executable instance of the computer program code.*

see pages 4-5, referring to Profiling and compiling heuristics.

As per Claim 53: Regarding,

> *The optimizing compiler of claim 34, encoded in one or more computer readable media selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.*

see rationale in Claim 1.

As per Claim 61: Regarding,

> *A computer program product encoded in one or more computer readable media, the computer program product comprising: a first execution sequence; and an information encoding associating an execution event with at least some operation of the first execution sequence, the associated execution event based at least in part on an execution profile of the first execution sequence of operations, wherein consistency of the association is maintained from preparation of the first executable instance for preparation of a second executable instance.*

see rationale in Claim 1.

As per Claim 62: Regarding,

The computer program product of claim 61, wherein the execution event is a cache miss likelihood.

see rationale in Claim 11.

As per Claim 63: Regarding,

The computer program product of claim 61, wherein the associated operation is a memory access operation.

see rationale in Claim 10.

As per Claim 64: Regarding,

The computer program product of claim 61, employed in an data structure of an optimizing compiler in preparation of an optimized instance of the execution sequence of operations, wherein the optimized instance includes one or more prefetch operations placed before particular ones of the memory access operations for which the associated information encoding indicates a cache miss likelihood.

see rationale in Claim 12.

As per Claim 65: Regarding,

The computer program product of claim 61, wherein the one or more computer readable media are selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.

see rationale in Claim 1.

As per Claim 66: Claim 66 recites an apparatus that performs the steps corresponding to the limitation of Claim 1. See the rationale on Claim 1 above.

As per Claim 67: Regarding,

The apparatus of claim 66, wherein the identifying includes producing a table of tags and operation addresses.

as part of profiling, in page 4, Profiling and compiling heuristics.

<u>As per Claim 68</u>: Regarding,

The apparatus of claim 66, wherein information for the identifying is encoded in a file or communications channel read by the means for collecting.

as part of profiling, in page 4, Profiling and compiling heuristics.

<u>As per Claim 69</u>: Regarding,

The apparatus of claim 66, further comprising: means for preparing the first executable instance of the computer program code.

as part of profiling, in pages 3-4, How the dynamic compiler work and Profiling and compiling heuristics.

Conclusion

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (571) 272-3706. The examiner can normally be reached on 8:00AM to 5:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3694. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

TTV
Patent Examiner
Art Unit 2192
May 24, 2005